# ConSTGAT: Contextual Spatial-Temporal Graph Attention Network for Travel Time Estimation at Baidu Maps

Xiaomin Fang, Jizhou Huang*, Fan Wang, Lingke Zeng, Haijin Liang, Haifeng Wang
Baidu Inc., China
{fangxiaomin01,huangjizhou01,wang.fan,zenglingke,lianghaijin,wanghaifeng}@baidu.com

## ABSTRACT

The task of travel time estimation (TTE), which estimates the travel time for a given route and departure time, plays an important role in intelligent transportation systems such as navigation, route planning, and ride-hailing services. This task is challenging because of many essential aspects, such as traffic prediction and contextual information. First, the accuracy of traffic prediction is strongly correlated with the traffic speed of the road segments in a route. Existing work mainly adopts spatial-temporal graph neural networks to improve the accuracy of traffic prediction, where spatial and temporal information is used separately. However, one drawback is that the spatial and temporal correlations are not fully exploited to obtain better accuracy. Second, contextual information of a route, i.e., the connections of adjacent road segments in the route, is an essential factor that impacts the driving speed. Previous work mainly uses sequential encoding models to address this issue. However, it is difficult to scale up sequential models to large-scale real-world services. In this paper, we propose an end-to-end neural framework named ConSTGAT, which integrates traffic prediction and contextual information to address these two problems. Specifically, we first propose a spatial-temporal graph neural network that adopts a novel graph attention mechanism, which is designed to fully exploit the joint relations of spatial and temporal information. Then, in order to efficiently take advantage of the contextual information, we design a computationally efficient model that applies convolutions over local windows to capture a route's contextual information and further employs multi-task learning to improve the performance. In this way, the travel time of each road segment can be computed in parallel and in advance. Extensive experiments conducted on large-scale real-world datasets demonstrate the superiority of ConSTGAT. In addition, ConSTGAT has already been deployed in production at Baidu Maps, and it successfully keeps serving tens of billions of requests every day. This confirms that ConSTGAT is a practical and robust solution for large-scale real-world TTE services.

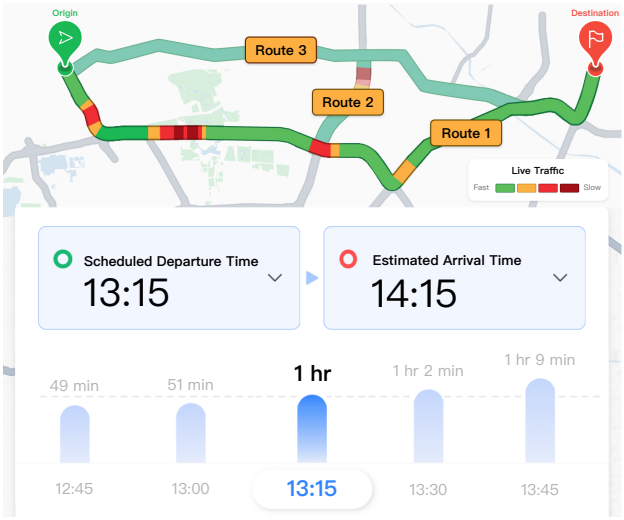



**Figure 1: A screenshot of the travel time estimation function at Baidu Maps mobile application.**

## CCS CONCEPTS

• **Information systems** → **Data mining**; • **Applied computing** → *Transportation.*

## KEYWORDS

Travel time estimation, graph neural network, attention mechanism, contextual information, transportation, Baidu Maps

**ACM Reference Format:**
Xiaomin Fang, Jizhou Huang, Fan Wang, Lingke Zeng, Haijin Liang, Haifeng Wang. 2020. ConSTGAT: Contextual Spatial-Temporal Graph Attention Network for Travel Time Estimation at Baidu Maps. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining USB Stick (KDD '20), August 23–27, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403320

## 1 INTRODUCTION

The task of travel time estimation (TTE), which estimates the travel time for a given route and departure time, plays an important role in intelligent transportation systems such as navigation, route planning, and ride-hailing services. TTE is an indispensable function of the navigation services in many mobile map applications such as Baidu Maps, which is one of the largest mobile map applications with over 340 million monthly active users worldwide by the end

*Corresponding author.


*KDD '20, August 23–27, 2020, Virtual Event, USA*

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403320

of December 2016.[1] Figure 1 shows an example of the travel time estimation function at Baidu Maps.[2] As illustrated in this example, it provides a user-friendly function that estimates the travel time in accordance with the choices of route and departure time, which greatly helps the drivers to know the traffic condition in advance and plan their trips wisely. For example, the driver departing from the origin could arrive at the destination within 1 hour by choosing the route 1 and the departure time 13:15. In order to help drivers to arrive at their destinations timely, it is important to develop a TTE module that is able to provide accurate and reliable travel time estimations.

TTE is challenging as there exists many aspects that could influence the accuracy of estimation, such as traffic prediction and contextual information of a route (e.g., traffic lights and turns). First, the accuracy of traffic prediction is strongly correlated with the traffic speed of the road segments in a route, especially of the road segments that are far from the origin. Existing approaches on traffic prediction mainly adopt spatial-temporal graph neural networks (STGNNs) [6, 7, 11, 22, 24] to model the road network, where spatial and temporal information is used separately. However, one drawback of them is that the spatial and temporal correlations are not fully exploited to obtain better accuracy. Second, the contextual information of a route, i.e., the connections of adjacent road segments in the route, is an essential factor that impacts the driving speed. For example, a driver usually spends different times on turning left and turning right. On one hand, the segment-based approaches [1, 17, 18] estimate each road segment of a given route independently, which are efficient as the travel time of each road segment can be computed in parallel and in advance. However, they ignore the contextual information of the route. On the other hand, the end-to-end approaches [10, 16, 19, 23] consider the contextual information of a route, which take all road segments in the route as a whole by applying sequence encoding structures. However, the main drawback of them is the high computation cost as the recurrent-based structures have to deal with the road segments in the route one-by-one in real time, and the situation would become worse when a route contains hundreds of road segments.

In this paper, we propose an end-to-end neural framework named ConSTGAT, which integrates traffic prediction and contextual information to address these two problems. Specifically, we first propose a spatial-temporal graph neural network that adopts a novel graph attention mechanism, which is designed to fully exploit the joint relations of spatial and temporal information. Then, in order to efficiently take advantage of the contextual information, we design a computationally efficient model that applies convolutions over local windows to capture a route's contextual information and further employs multi-task learning to improve the performance. ConSTGAT also takes insights from the segment-based approaches, which is able to preserve high inference efficiency by computing the travel time of each road segment in parallel and in advance.

To verify the effectiveness of the proposed framework ConSTGAT, we have conducted extensive experiments on large-scale real-world datasets collected from Baidu Maps. The experimental results demonstrate that ConSTGAT significantly outperforms the mainstream approaches in terms of multiple metrics. In addition, it has already been deployed in production at Baidu Maps, which successfully keeps serving tens of billions of requests every day.

Our main contributions can be summarized as follows:

- **Potential impact**: We propose an end-to-end neural framework as an industrial solution to the travel time estimation function in mobile map applications. It is our first attempt to deploy the framework into Baidu Maps to serve tens of billions of requests every day.
- **Novelty**: The design and implementation of this framework are driven by the novel ideas that fully exploit the joint relations of spatial and temporal information with a spatial-temporal graph attention network, as well as efficiently take advantage of a route's contextual information by applying convolutions over local windows and multi-task learning.
- **Technical quality**: Extensive experiments conducted on large-scale real-world datasets demonstrate the superiority of the proposed framework. The successful deployment of ConSTGAT at Baidu Maps further shows that it is a practical and robust solution for large-scale real-world TTE services.

## 2 RELATED WORK

### 2.1 Travel Time Estimation

The major categories of TTE are segment-based methods and end-to-end methods.

The segment-based methods [1, 17, 18] predict the travel time of each road segment in a route independently and then sum up all predicted times to get the total travel time of the route. These methods are widely used in navigation services, as they are computationally efficient in that the travel time of each road segment can be calculated in parallel and in advance. Although the segment-based approaches are efficient, they ignore the contextual information of the route, especially for the junction of two adjacent road segments.

The end-to-end methods [16, 19, 23] take a route as a whole and estimate the travel time of the route directly. They mainly employ recurrent structures to capture the contextual information (e.g., traffic lights and turns) of the route. The end-to-end methods are more effective than the segment-based methods. Wang et al. [16] consider the relations between road segments in a route by convolution and stacked long-short term memory neural network (LSTM). Similarly, Zhang et al. [23] leverage bi-directional LSTM to capture the contextual information. Wang et al. [19] propose a Wide-Deep-Recurrent model that combines the Wide&Deep model and recurrent models, where LSTM is used to capture the contextual information of the route. Although recurrent structures are capable of learning the correlations between the road segments in a route, their computation costs are too expensive for large-scale navigation services. A route may contain hundreds or even thousands of road segments, but the hidden states of the road segments in the route are computed in sequence and cannot be computed in advance. Thus, end-to-end methods are hard to scale up.

The proposed framework ConSTGAT takes advantage of both segment-based methods and end-to-end methods, in which the travel times of the road segments in a route are estimated simultaneously. To obtain the contextual information, we leverage the

[1] http://ir.baidu.com/static-files/e249a0f8-082a-4f8a-b60d-7417fa2f8e7e
[2] We translate the example from Chinese to English for the sake of understanding.

convolution structure and use the loss functions of the road segments as well as the entire route.

## 2.2 Spatial-Temporal Graph Neural Networks

Recently, there is increasing interest in graph neural networks (GNNs) [5, 13, 20]. Lots of studies have applied GNNs to graph-based problems such as computer vision, natural language processing, recommender systems, and other domains by employing the graph structures.

The rise of GNNs inspired the following work to employ spatial-temporal GNNs (STGNNs) on traffic prediction [6, 7, 11, 21, 22, 24]. Compared with the canonical GNNs, STGNNs leverage both spatial and temporal information. A typical STGNN first encodes the geographic information by graph convolution network (GCN) [3] or graph attention network (GAT) [14], and then encodes the temporal information with LSTM, CNN, or attention mechanism. Although these studies can make use of spatial and temporal information, they assume that the relations of geographic information and temporal information are independent and do not consider the joint relations of them. To address this problem, we propose a novel graph neural network that encodes the geographic and temporal information simultaneously, which is designed to fully capture the correlations of spatial and temporal information.

# 3 PRELIMINARY

In this section, we formalize the problem of TTE, and then introduce the features we used in the proposed framework.

## 3.1 Problem Definition

**Road network**: The road network is an essential component for estimating the travel time of a given route. In this study, we define the road network as a directed graph $\mathcal{G} = (\mathcal{L}, \mathcal{E})$, where $\mathcal{L}$ is a link set and $\mathcal{E}$ is an edge set. Link $l \in \mathcal{L}$ represents a road segment. For the sake of convenience, "road segment" is referred to as "link" hereafter. Edge $e_{ij} \in \mathcal{E}$ denotes the edge connecting link $l_i$ and link $l_j$, if link $l_i$ and link $l_j$ share the same junction.

**Route**: A route $r$ is a link sequence $r = [l_1, l_2, \cdots, l_m]$, where $m$ is the number of links in that route. Usually, a route contains dozens of or hundreds of links. A navigation service produces several candidate routes based on the corresponding road network.

**Request**: A request is represented by a pair $req = (r, s)$, where $r$ is the route, and $s$ is the departure time of the route. The task of TTE is to estimate the travel time $y$ of a request $req$.

**Dataset**: A dataset is defined as $\mathcal{D} = \{(req^{(i)}, y^{(i)})\}_{i=1}^{n}$, where $y^{(i)}$ is the ground-truth travel time for request $req^{(i)}$, and $n$ is the number of requests in the dataset. For $(req^{(i)}, y^{(i)}) \in \mathcal{D}$, the travel time of the $j$-th link $l_j^{(i)}$ in route $r^{(i)}$ of request $req^{(i)}$ is denoted as $y_j^{(i)}$, which is computed as $y^{(i)} = \sum_{j=1}^{m^{(i)}} y_j^{(i)}$.

## 3.2 Feature Extraction

We extract features from the road network, historical traffic conditions, and background information for TTE.

**Road network**: The road network describes the relations between the links. The following information of each link is extracted as the features for use: the id, the length, the width, the number of lanes, the type, the speed limit, the type of crossing, and the kind of traffic light. Moreover, the graph structure of the road network is employed to describe geographic relations.

**Historical traffic conditions**: As the historical traffic conditions have a significant effect on traffic prediction, several types of traffic speeds at different time slots are taken as features. For example, the median travel speed and the mean travel speed of a given link $l$ at time slot $t$ are deduced from the traffic records in the dataset, where a time slot is set to 5 minutes in this paper. Moreover, the uncertainty of the collected historical traffic speeds also affects the accuracy of traffic prediction. For this reason, the number of collected travel speeds is leveraged.

**Background information**: TTE could be affected by lots of background information. The departure time is one of the most important factors. The rush hours, weekdays, and other time-related information are also extracted as features. We denote the background information of the $i$-th link in the route as $x_i^{(B)}$.

# 4 ConSTGAT

In this section, we detail the proposed end-to-end neural framework.

## 4.1 Framework

Figure 2 shows the architecture of the proposed framework ConSTGAT, which consists of three modules: *Contextual Information*, *Traffic Prediction*, and *Integration*. *Traffic Prediction* module adopts a novel spatial-temporal graph attention network to capture the joint relations of spatial and temporal information of the traffic conditions. *Contextual Information* module and *Integration* module focus on addressing the problem of contextual information of the route, where *Contextual Information* module utilizes convolution structures to capture the relations of the adjacent links, and *Integration* module employs multi-task learning to improve the performance.

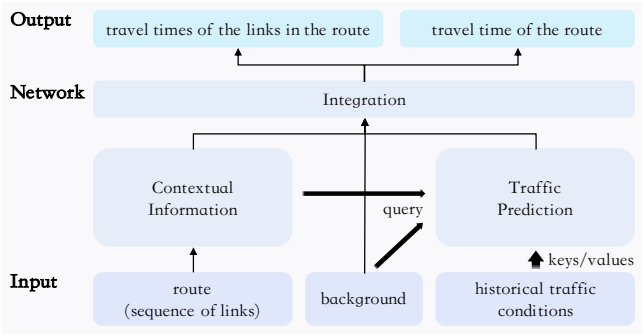



**Figure 2: Architecture of ConSTGAT.**

## 4.2 Traffic Prediction

A link's future traffic condition is strongly correlated with the historical traffic conditions of the link itself as well as its neighbor links. For example, traffic congestion on a link could have a high probability of causing traffic congestion on its neighbor links in a short time. Although there is a growing interest in predicting traffic with spatial-temporal graph neural networks, where spatial and temporal information is used separately. To fully exploit the

joint relations of spatial and temporal information, we propose a novel spatial-temporal graph attention network, which deals with the geographic information and temporal information of the traffic conditions simultaneously. We also introduce masking mechanism to improve the robustness of the proposed model.

We denote the traffic conditions observed on graph $\mathcal{G}$ at time slot $t$ as $C^t$ and the traffic conditions observed on link $l$ as $c_l^t$. *Traffic Prediction* module takes historical traffic conditions, graph $\mathcal{G}$, and the departure time $s$ as inputs to predict the future traffic conditions of graph $\mathcal{G}$:

$$[C^{s-T_h}, \cdots, C^{s-1}; \mathcal{G}] \rightarrow [\hat{C}^s, \hat{C}^{s+1}, \cdots, \hat{C}^{s+T_f-1}], \tag{1}$$

where $T_f$ is the number of the predicted future time slots, while $T_h$ is the number of the historical time slots used to train the model.

We detail the 3D-attention mechanism used to capture the spatial-temporal relations for traffic conditions, which is an extension of graph attention network [15], and is illustrated in Figure 3.

First, we extract a spatial-temporal tensor. Given graph $\mathcal{G} = (\mathcal{L}, \mathcal{E})$, the neighbor set of link $l_i \in \mathcal{L}$ is defined as $\mathcal{NB}(l_i) = \{l_j | e_{ij} \in \mathcal{E}\}$. We assume that the future traffic condition of link $l_i$ is directly affected by the historical traffic conditions of itself and the links nearby. The historical traffic conditions can be organized into a matrix $X_i^{(ST)} \in \mathbb{R}^{|\mathcal{NB}(l_i)|T_h \times d^{(ST)}}$, where $|\mathcal{NB}(l_i)|$ is the number of neighbors for link $l_i$, $T_h$ is the number of historical time slots, and $d^{(ST)}$ is the dimension of the features. To be specific, $X^{(ST)}_{i,(j-1)T_h+k}$ represents the traffic condition of the $k$-th time slot of the $j$-th neighbor of link $i$ with $j \in [1, \mathcal{NB}(l_i)]$ and $k \in [1, T_h]$. Moreover, the features of the neighbor links (e.g., id and lane number) are organized as matrix $X_i^{(S)} \in \mathbb{R}^{|\mathcal{NB}(l_i)| \times d^{(S)}}$. The features of the historical time slots are organized as matrix $X^{(T)} \in \mathbb{R}^{T_h \times d^{(T)}}$. Here, $d^{(S)}$ and $d^{(T)}$ are the dimensions of the features for the spatial and temporal information, respectively. Then, matrices $X_i^{(ST)}, X_i^{(S)}$, and $X^{(T)}$ are merged into a new spatial-temporal matrix $X_i^{(MST)} \in \mathbb{R}^{|\mathcal{NB}(l_i)|T_h \times d^{(MST)}}$ with $d^{(MST)} = d^{(ST)} + d^{(S)} + d^{(T)}$ by expanding matrices $X_i^{(S)}$ and $X^{(T)}$:

$$\begin{aligned} X^{(MST)}_{i,j,k} = \textbf{Concat}(X^{(ST)}_{i,(j-1)T_h+k}, X^{(S)}_{i,j}, X^{(T)}_{k}), \\ j \in [1, \mathcal{NB}(l_i)], k \in [1, T_h], \end{aligned} \tag{2}$$

where **Concat** denotes a concatenation operation. Matrix $X_i^{(MST)} \in \mathbb{R}^{|\mathcal{NB}(l_i)|T_h \times d^{(MST)}}$ can be reshaped into a 3D-tensor $X_i^{(MST)} \in \mathbb{R}^{|\mathcal{NB}(l_i)| \times T_h \times d^{(MST)}}$, which is used as the spatial-temporal tensor (abbr. ST-tensor) for the traffic conditions of the $i$-th link.

Then, we use attention mechanism [2] to obtain the relations between the traffic conditions. We introduce a novel 3D-attention setting to capture the joint relations of spatial and temporal information. In this setting, the combination of the contextual information $x_{i,w}^{(CI)}$ and the background information $x_i^{(B)}$ is taken as the *query* of the attention mechanism. The details of $x_{i,w}^{(CI)}$ will be described in the following subsection. The vectors in ST-tensor $X^{(MST)}_{i,j,k} (j \in [1, |\mathcal{NB}(l_i)|], k \in [1, T_h])$ are taken as the *keys* and *values* of the attention mechanism. To be specific, the 3D-attention mechanism is formulated as

$$Q_i = \textbf{Dense}(\textbf{Concat}(x_{i,w}^{(CI)}, x_i^{(B)})), \tag{3}$$

$$K_{i,j,k} = \textbf{Dense}(X^{(MST)}_{i,j,k}), \tag{4}$$

$$V_{i,j,k} = \textbf{Dense}(X^{(MST)}_{i,j,k}), \tag{5}$$

$$f(Q_i, K_{i,j,k}) = \frac{Q_i^T \cdot K_{i,j,k}}{\sqrt{d^{(H)}}}, \tag{6}$$

$$\alpha(Q_i, K_{i,j,k}) = \frac{exp(f(Q_i, K_{i,j,k}))}{\sum_{j',k'} exp(f(Q_i, K_{i,j',k'}))}, \tag{7}$$

$$Attention(Q_i, K_i, V_i) = \sum\nolimits_{j,k} \alpha(Q_i, K_{i,j,k}) V_{i,j,k}, \tag{8}$$

where **Dense** denotes a fully-connected layer with an activation function, and $d^{(H)}$ denotes the hidden size of the attention mechanism. Then, the relations between *query* and the historical traffic condition for link $l_i$ can be encoded as $x_i^{(TC)} = Attention(Q_i, K_i, V_i)$ by Equation (8). For this reason, we call the proposed graph neural network by 3D-attention mechanism "3DGAT".

It is worth noting that 3DGAT can be employed to other spatial-temporal applications as well. Compared with the vanilla spatial-temporal graph neural networks, 3DGAT is elaborately designed to fully exploit the joint relations of spatial and temporal information.

Sometimes, the traffic conditions may get lost due to weak network signals or other reasons. To alleviate this problem, we employ masking mechanism to improve the robustness of the model. Masking mechanism has been proven to be effective for improving the task of language model pre-training in natural language processing [4]. In the training phase, we assume some of the traffic conditions are unknown to the model. To this end, we randomly mask 10% of the historical traffic conditions. This means for request $req = (r, s)$, 10% of $c_i^t$ with $i \in [1, m]$ and $t \in [s - T_h, s - 1]$ are set to zero vectors. From the perspective of model training, adding noise to a neural network can reduce overfitting and improve model generalization. In the test phase, we use all the historical traffic conditions to evaluate the performance of our model.

## 4.3 Contextual Information

We develop a computationally efficient method that applies convolutions to capture the contextual information in *Contexual Information* module, as well as leverages multi-task learning in *Integration* module. The proposed method takes insights from the segment-based methods, which is able to efficiently reduce the response time of a request. Figure 4 shows the architecture of this method.

The contextual information of a route, such as the angle of two adjacent links, the relationship between the main road and the auxiliary road, plays an important role in estimating the travel time of the route. To capture the contextual information, we estimate the travel time of each link by employing the information of other links in the given route. To do this efficiently, we only consider the nearby links of a link $l_i$ in route $r$ rather than all the links in $r$, which is based on the observation that the remote links often have markedly less impact than the nearby ones. To represent the nearby links, we introduce sub-path $p_{i,w} = [l_{i-w}, \cdots, l_{i-1}, l_i, l_{i+1}, \cdots, l_{i+w}]$ with contextual window size $w$. If window size $w = 0$, only the information of link $l_i$ is used, which is equivalent to the setting of the

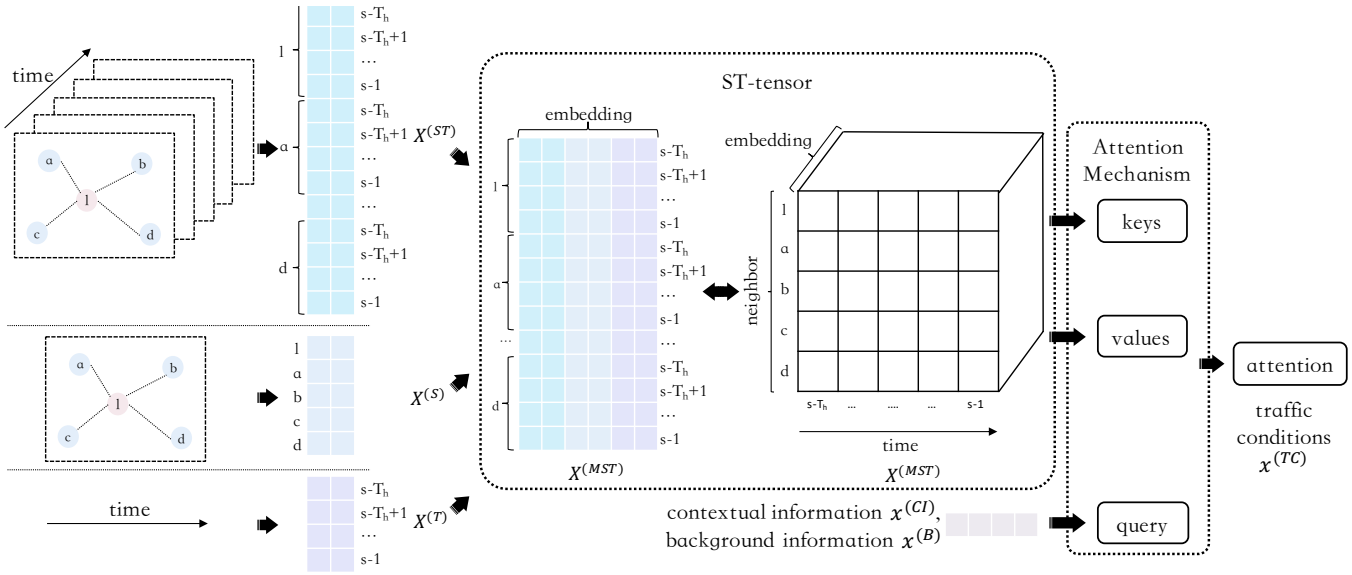



**Figure 3: 3D-attention mechanism for spatial-temporal graph attention network.**

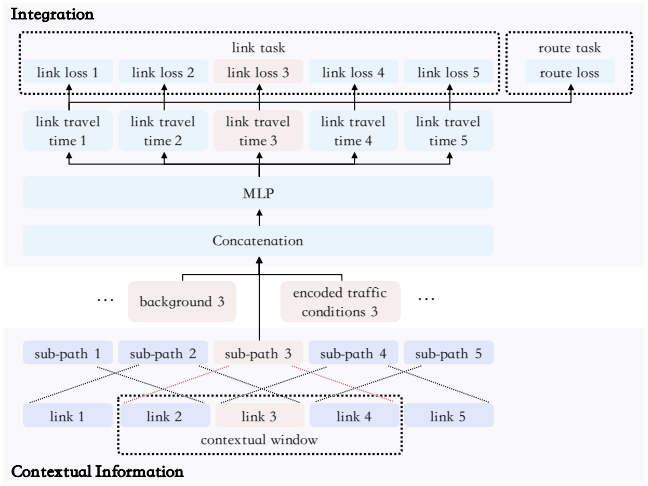



**Figure 4: Architecture of *Contextual Information* module and *Integration* module.**

segment-based methods. If window size $w \to \infty$, all the links in route $r$ are taken into account, which is equivalent to the setting of the vanilla end-to-end methods.

Convolutional neural network (CNN) [9] is an efficient method to obtain the local dependencies. Thus, we use a convolutional layer to encode local dependency, which is formulated as

$$x_{i,w}^{(CI)} = \textbf{Dense}(\textbf{Concat}(\textbf{Emb}(l_{i-w}), \cdots, \textbf{Emb}(l_i), \cdots, \textbf{Emb}(l_{i+w}))), \quad (9)$$

where **Emb** denotes a representation layer. The information of sub-path $p_{i,w}$ with window size $w$ for link $l_i$ is encoded as $x_{i,w}^{(CI)} \in \mathbb{R}^{d^{(CI)}}$, where $d^{(CI)}$ represents the dimension of the contextual information. The encoded contextual information of link $l_i$ serves as the inputs of *Traffic Prediction* module and *Integration* module.

As shown in Figure 4, *Integration* module is responsible for aggregating the outputs of the contextual information $x_{i,w}^{(CI)}$, the traffic conditions $x_i^{(TC)}$, and the background information $x_i^{(B)}$. It first employs a concatenation layer and then multiple fully-connected layers [12] to collect all the information needed to predict the travel times of the links and the given route. Given request $req = (r, s)$, the travel time of link $l_i$ in route $r$ is predicted by

$$\hat{y}_i = \textbf{MLP}(\textbf{Concat}(x_{i,w}^{(CI)}, x_i^{(B)}, x_i^{(TC)})), \quad (10)$$

where **MLP** represents the multiple fully-connected layers.

Then, the predicted travel times of the links in the route are summed up to obtain the predicted travel time of the route:

$$\hat{y} = \sum_{i=1}^{m} \hat{y}_i, \quad (11)$$

where $\hat{y}$ denotes the predicted travel time of the route.

To further improve the performance, we combine the segment-based methods and the vanilla end-to-end methods by employing the loss functions of both the links and the entire route.

Huber loss [8] is a widely used loss function for regression problems, which is able to alleviate the impact of the outliers in comparison with square loss. It is used as the loss of the links $L_{link}$, which is defined as

$$L_{link}(\hat{y}_i, y_i) = \begin{cases} \frac{1}{2}(\hat{y}_i - y_i)^2 & |\hat{y}_i - y_i| < \delta, \\ \delta(|\hat{y}_i - y_i| - \frac{1}{2}\delta) & \text{otherwise}, \end{cases} \quad (12)$$

where $\delta$ is a hyper-parameter.

We use the absolute percentage error (APE) as the loss function of the entire route $L_{route}$, which is defined as

$$L_{route}(\hat{y}, y) = \frac{|y - \hat{y}|}{y}. \quad (13)$$

Then, we combine the loss functions of the links $L_{link}$ and the route $L_{route}$:

$$L = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{m^{(i)}}\sum_{j=1}^{m^{(i)}} L_{link}(\hat{y}_j^{(i)}, y_j^{(i)}) + L_{route}(\hat{y}^{(i)}, y^{(i)})). \quad (14)$$

It is equivalent to synthesize the segment-based methods and end-to-end methods. The model is trained by minimizing the objective function $L$.

The navigation service in Baidu Maps needs to serve tens of billions of requests every day. Therefore, the response time of a request is of great importance to user experience. However, existing methods are hard to scale up since they mainly adopt sequence encoding models, which do the computation in real time.

To reduce the response time of the service, we borrow the ideas from the segment-based methods and elaborately design the prediction process. The prediction process consists of two steps. In the first step, we predict the travel times of all the links in $\mathcal{L}$ in different situations and save the predicted travel times in a look-up table. Specifically, for link $l \in \mathcal{L}$, we estimate the travel times of link $l$ with different sub-paths $\mathcal{P}(l)$ and different departure time slots $\mathcal{T}$. Here, $\mathcal{P}(l)$ denotes the candidate sub-paths containing link $l$, and $\mathcal{T} = \{s, s+1, s+2, \cdots, s+T_f-1\}$ denotes the candidate departure time slots, where $T_f$ is the number of candidate departure time slots. All the travel times can be computed in parallel. In the second step, when a request arrives, we look up the table that saves the travel times to find the travel times of the corresponding links with the situations, and then we sum up these travel times. In this way, we can compute the travel times of the links with different situations in advance and in parallel. Therefore, it is easy to deploy the proposed method to large-scale real-world navigation services.

# 5 EXPERIMENTS

## 5.1 Experimental Settings

We compare ConSTGAT with several strong baseline methods using real-world datasets. We collect a large number of routes and road networks of the cities Taiyuan, Hefei, and Huizhou. The datasets are sampled from the logs of Baidu Maps for the period spanning from July 21st to Aug 31st, 2019. The data of the first four weeks are used for model training, while the data of the last week are used for evaluation. Table 1 shows the statistics of the datasets used in the experiments.

We use three metrics including mean absolute percentage error (MAPE), mean average error (MAE), and root mean square error (RMSE) to evaluate all methods, which are widely used to evaluate the accuracy of regression problems. They are defined as

$$\text{MAPE}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{n}\sum_{i=1}^{n}\frac{|y^{(i)} - \hat{y}^{(i)}|}{y^{(i)}},$$

$$\text{MAE}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{n}\sum_{i=1}^{n}|y^{(i)} - \hat{y}^{(i)}|,$$

$$\text{RMSE}(\hat{y}^{(i)}, y^{(i)}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2}.$$

We set the embedding sizes of all attributes to 8. In *Traffic Prediction* module, we adopted the traffic conditions at $T_h = 12$ historical time slots and set the hidden size of the 3D-attention mechanism $d^{(H)} = 32$. In *Contextual Information* module, we set the number of the filters to 32 and the default window size $w = 1$. In *Integration* module, we employ two-layer MLP, in which the output size of the first fully-connected layer is set to 64. We predicted the travel times of the links at $T_f = 12$ future time slots.

**Table 1: Statistics of datasets for training and test.**

| Dataset | Links | Training routes | Test routes | Average links per route |
|---|---|---|---|---|
| Taiyuan | 216,208 | 3,207,830 | 876,580 | 62.16 |
| Hefei | 376,310 | 4,571,573 | 851,188 | 55.36 |
| Huizhou | 346,396 | 4,153,568 | 727,492 | 52.90 |

## 5.2 Methods for Comparison

We compare the proposed ConSTGAT with five baselines:

- **AVG**. We calculate the average traffic speeds of 2016 (=7 weekdays × 24 hours × 12) time slots in each city to estimate the travel times of links with different departure time slots. Given a request $req = (r, s)$, we sum up the corresponding estimated travel times of all the links in the route $r$.
- **DeepTravel** [23]. This is an end-to-end method. It first extracts spatial and temporal features and then employs bidirectional LSTM and auxiliary tasks based on duel intervals.
- **STANN** [7]. This is a spatial-temporal graph neural network. It first encodes the spatial information by graph attention, and then encodes the temporal information by LSTM and attention mechanism. However, standard STANN is not practical for our task because it considers the relations of all the links in the graph for each link. In our experiments, we only consider the relations to its neighbor links.
- **DCRNN** [11]. This is also a spatial-temporal graph neural network. It first captures the spatial dependency by graph convolution network (a special case of diffusion convolution network), and then captures temporal dependency by LSTM.
- **GAT+Attention**. To verify whether the proposed network 3DGAT is superior to existing spatial-temporal networks, we develop a model named GAT+Attention to conduct an ablation test. This new model first aggregates the spatial information by graph attention network, and then aggregates the temporal information by attention mechanism. 3DGAT and GAT+Attention differ only in one modality: 3DGAT considers the spatial and temporal information simultaneously, while GAT+Attention considers them separately.

Among these methods, STANN, DCRNN, and GAT+Attention employ graph neural networks to estimate the travel times of the links rather than that of an entire route. To make them comparable, we use the same setting of *Integration* module in ConSTGAT, which leverages multi-task learning to optimize the model. In addition, DeepTravel, STANN, DCRNN, and GAT+Attention use the similar parameter settings as used by ConSTGAT. For DeepTravel, the hidden size of the bidirectional LSTM is set to 32. For STANN, DCRNN, and GAT+Attention, the hidden sizes of the graph neural

**Table 2: Performance of ConSTGAT and other methods for estimating the travel times of the routes.**

| Method | Taiyuan | | | Hefei | | | Huizhou | | |
|---|---|---|---|---|---|---|---|---|---|
| | MAPE | MAE (sec) | RMSE (sec) | MAPE | MAE (sec) | RMSE (sec) | MAPE | MAE (sec) | RMSE (sec) |
| **AVG** | 39.86% | 330.39 | 607.59 | 54.26% | 466.52 | 872.41 | 52.65% | 352.14 | 750.63 |
| **DeepTravel** | 24.25% | 156.65 | 309.56 | 32.10% | 160.68 | 327.75 | 34.01% | 157.85 | 354.14 |
| **STANN** | 24.80% | 167.04 | 319.94 | 32.16% | 172.04 | 354.38 | 32.65% | 159.92 | 362.05 |
| **DCRNN** | 24.28% | 163.25 | 315.74 | 30.57% | 162.28 | 341.73 | 30.97% | 149.70 | 347.11 |
| **GAT+Attention** | 22.94% | 142.98 | 283.55 | 26.63% | 132.46 | 297.30 | 30.38% | 142.47 | 334.48 |
| **ConSTGAT** | **21.79%** | **130.48** | **259.89** | **25.99%** | **127.06** | **289.56** | **27.10%** | **118.87** | **291.81** |

**Table 3: Performance of different spatial-temporal graph neural networks for estimating the travel times of the links.**

| Method | Taiyuan | | | Hefei | | | Huizhou | | |
|---|---|---|---|---|---|---|---|---|---|
| | MAPE | MAE (sec) | RMSE (sec) | MAPE | MAE (sec) | RMSE (sec) | MAPE | MAE (sec) | RMSE (sec) |
| **STANN** | 52.21% | 5.36 | 24.17 | 61.62% | 7.37 | 33.19 | 63.37% | 6.65 | 33.45 |
| **DCRNN** | 55.53% | 5.38 | 24.05 | 61.00% | 6.98 | 32.76 | 63.00% | 6.40 | 33.14 |
| **GAT+Attention** | 47.61% | 4.76 | 23.13 | 46.91% | 5.76 | 30.96 | 52.39% | 6.01 | 32.66 |
| **3DGAT** | **46.08%** | **4.42** | **22.11** | **46.15%** | **5.62** | **30.66** | **48.72%** | **5.25** | **31.80** |

networks are set to 32, while the hidden sizes of the LSTMs and attention mechanisms are set to 32.

## 5.3 Experimental Results

We implemented all the methods except AVG by *PaddlePaddle*[3], an open-source deep learning platform maintained by Baidu. We report empirical results and analysis in this subsection.

*5.3.1 Overall Evaluation.* We compare ConSTGAT with several baseline methods on three real-world datasets. Table 2 shows the experimental results. Boldface indicates the best score w.r.t. each metric. From the results, we have the following observations. First, ConSTGAT significantly outperforms other methods on all datasets. AVG is a simple baseline and works the worst. The end-to-end method DeepTravel performs better as it considers the contextual information, but it does not consider the graph structures of the road networks for traffic prediction. Second, among the STGNN based methods (i.e., STANN, DCRNN, GAT+Attention, and ConSTGAT) that leverage the graph structures, ConSTGAT significantly outperforms other methods because it can fully exploit the joint relations of spatial and temporal information.

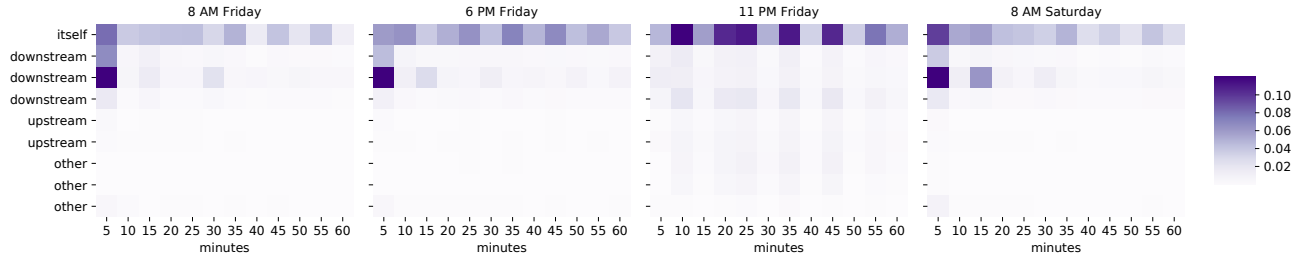
*5.3.2 Spatial-Temporal Graph Neural Networks.* 3DGAT is a spatial-temporal neural network that can be deployed to other spatial-temporal applications as well. To verify the superiority of 3DGAT, we compare it with other STGNNs, including STANN, DCRNN, and GAT+Attention.

As a general STGNN focuses on estimating the traffic of the links rather than routes, we also compare the performance of STGNNs on estimating the travel times of the links. 3DGAT can be seen as a special version of ConSTGAT with the contextual window size $w = 0$. From the results in Table 3, we can see that 3DGAT is superior to other STGNNs for the problem of traffic prediction.

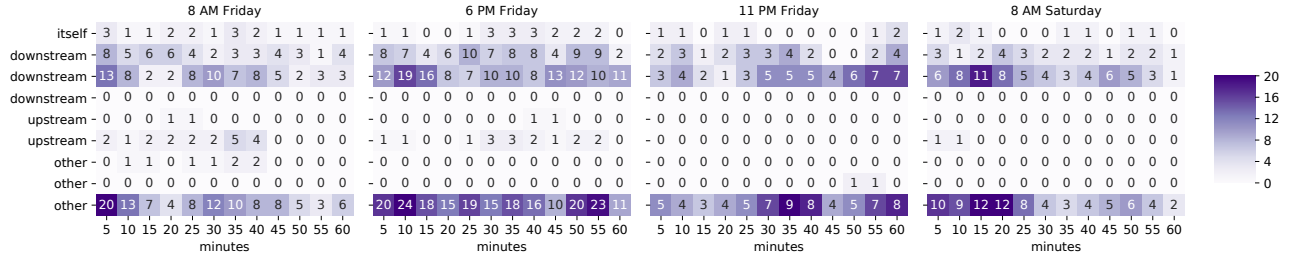
[3]The official site is at https://www.paddlepaddle.org.cn/.

To analyze the traffic correlations between spatial and temporal information, we draw the matrices of the attention weights. Figure 5 shows multiple cases with different situations. Figure 5a, Figure 5b, and Figure 5c present the matrices of the attention weights calculated by 3DGAT, the matrices of the number of the traffic records, and the matrices of the median historical travel times, respectively. We compare multiple cases on a specific link $l$ with nine neighbors (including link $l$ itself). Each row of the matrices indicates a neighbor link, while each column of the matrices indicates a historical time slot of 5 minutes. For the rows, the 1st row of each matrix represents the impact of link $l$ itself, and the 2nd to the 9th rows of each matrix represent the impacts of the adjacent links, where the 2nd to the 4th rows denote the downstream links and the 5th to the 6th rows denote the upstream links. For the columns, the first column indicates the latest time slot, while the last column indicates the furthest time slot. In Figure 5a, the darker the grid cell, the more relevant the corresponding neighbor link is to link $l$. Figure 5b and Figure 5c are similar. The figures show different spatial-temporal relations in different situations (e.g., rush hours): (1) the downstream links have greater relevance to link $l$ than other adjacent links; and (2) the link with more traffic records or larger historical travel times plays a more critical role.

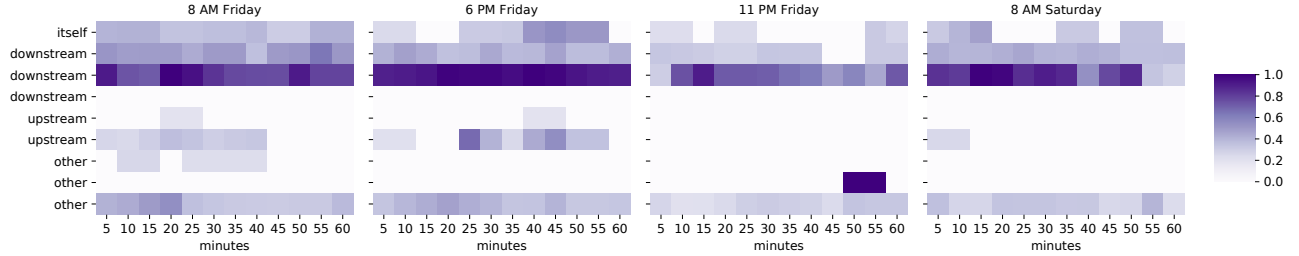
*5.3.3 Contextual Information.* We analyze the effect of the contextual information of a route. Figure 6 shows the performance comparison of ConSTGAT with different contextual window sizes for estimating the travel times of the links. A model with window size $w = 0$ is equivalent to a segment-based method, which indicates that no contextual information is leveraged. When window size $w > 0$, given link $l$ in a route, the information of links nearby in the route is used to take advantage of the contextual information. The experimental results show that the contextual information indeed helps to improve the performance of the models. The RMSE results of the models are sensitive to window sizes on datasets of Taiyuan




**(a) The matrices of the attention weight calculated by 3DGAT in different situations.**




**(b) The matrices of the number of traffic records in different situations.**




**(c) The matrices of the median historical travel times in different situations.**

**Figure 5: Case analysis of 3DGAT in different situations.**

and Huizhou. However, the results of the models with different window sizes are almost the same on dataset of Hefei.

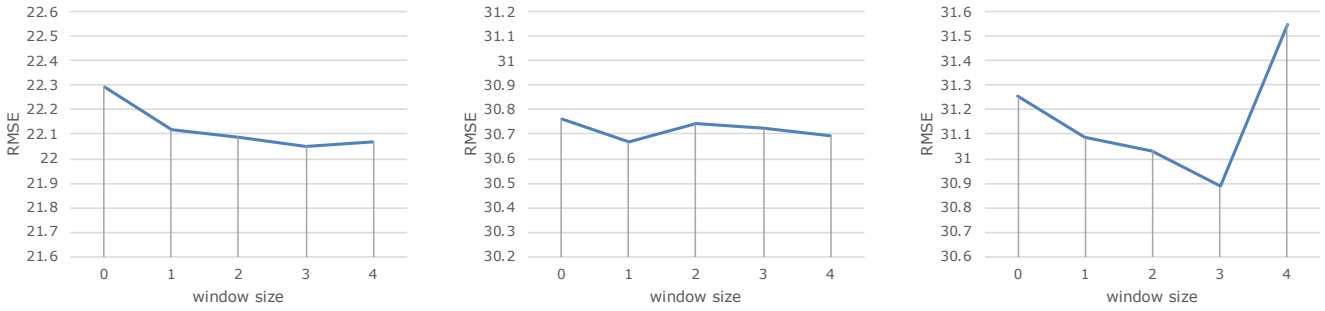
*5.3.4 Robustness.* We conduct experiments to validate whether the masking mechanism could improve the robustness of the models. The model with *train mask rate=10%* denotes a ConSTGAT model that randomly masks 10% of traffic conditions in the training phase, and *train mask rate=0%* denotes a model without masking. We assume some of the traffic conditions are lost and unknown to the models in evaluation. To this end, we randomly mask 0% to 100% traffic conditions of the test data, which is equivalent to add some noise. When *test mask rate=100%*, all the historical traffic conditions are unknown, and only the contextual information and the background information are left. From the results in Figure 7, we can see that the model with *train mask rate=10%* performs better than that with *train mask rate=0%* on all datasets. Moreover, the gaps between the two models on datasets Taiyuan and Hefei become larger with the increase of *test mask rate*. Although the effect of the masking mechanism is weaker on dataset Huizhou, the model with *train mask rate=10%* outperforms that with *train mask rate=0%* in overall performance. This demonstrates that it can improve the robustness of the models by employing masking mechanism in model training.

## 6 CONCLUSION

In this paper, we propose a computationally efficient end-to-end framework named ConSTGAT for the task of travel time estimation, which focuses on addressing the problems of traffic prediction and contextual information. To improve the accuracy of traffic prediction, we develop a novel STGNN to obtain the joint relations of spatial and temporal information by 3D-attention mechanism. Furthermore, in order to efficiently take advantage of the contextual information, we design a computationally efficient model that applies convolutions over local windows to capture a route's contextual information and further employs multi-task learning to improve the performance. In this way, we can take insights from the segment-based approaches, which is able to preserve high inference efficiency by computing the travel time of each link in parallel and in advance. Extensive experiments conducted on large-scale real-world datasets demonstrate the superiority of the proposed framework. The successful deployment of ConSTGAT at Baidu Maps further shows that it is a practical and robust solution for large-scale real-world services.

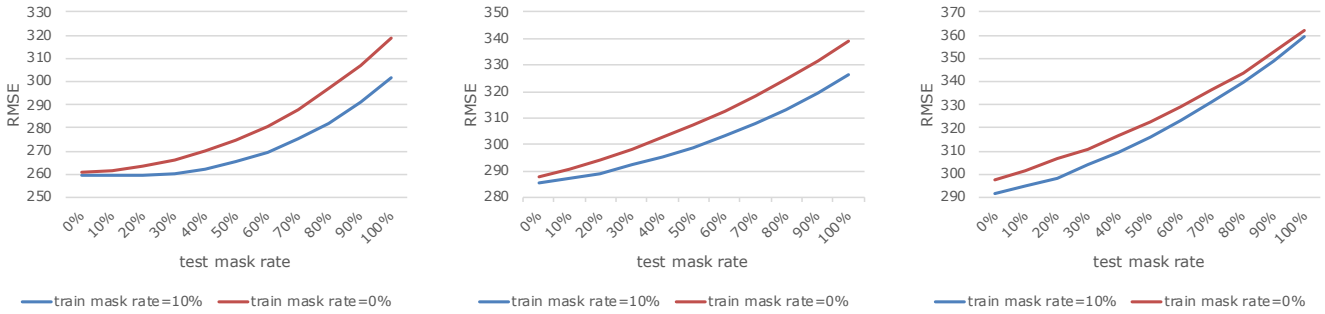
## REFERENCES

[1] Pouria Amirian, Anahid Basiri, and Jeremy Morley. 2016. Predictive analytics for enhancing travel time estimation in navigation apps of Apple, Google, and




(a) Taiyuan (b) Hefei (c) Huizhou

**Figure 6: Performance of ConSTGAT with different contextual window sizes for estimating the travel times of the links.**




(a) Taiyuan (b) Hefei (c) Huizhou

**Figure 7: Performance of ConSTGAT with different train and test mask rates for estimating the travel times of the routes.**

Microsoft. In *Proceedings of SIGSPATIAL*. 31–36.
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*.
[3] Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of NIPS*. 3844–3852.
[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*. 4171–4186.
[5] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *Proceedings of IJCNN*. 729–734.
[6] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. In *Proceedings of AAAI*. 922–929.
[7] Zhixiang He, Chi-Yin Chow, and Jia-Dong Zhang. 2018. STANN: A Spatio–Temporal Attentive Neural Network for Traffic Prediction. *IEEE Access* 7 (2018), 4795–4806.
[8] Peter J. Huber. 1964. Robust Estimation of a Location Parameter. *Annals of Mathematical Statistics* 35, 1 (1964), 73–101.
[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of NIPS*. 1097–1105.
[10] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task representation learning for travel time estimation. In *Proceedings of KDD*. 1695–1704.
[11] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *Proceedings of ICLR*.
[12] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. *Learning internal representations by error propagation*. Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science.
[13] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.
[14] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of ICLR*. https://openreview.net/forum?id=rJXMpikCZ
[15] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of ICLR*.
[16] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. In *Proceedings of AAAI*. 2500–2507.
[17] Hongjian Wang, Xianfeng Tang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2019. A Simple Baseline for Travel Time Estimation Using Large-scale Trip Data. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2, Article 19 (2019), 22 pages.
[18] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of KDD*. 25–34.
[19] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to estimate the travel time. In *Proceedings of KDD*. 858–866.
[20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596* (2019).
[21] Bing Yu, Mengzhang Li, Jiyong Zhang, and Zhanxing Zhu. 2019. 3D Graph Convolutional Networks with Temporal Graphs: A Spatial Information Free Framework For Traffic Forecasting. *arXiv preprint arXiv:1903.00919* (2019).
[22] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of IJCAI*. 3634–3640.
[23] Hanyuan Zhang, Hao Wu, Weiwei Sun, and Baihua Zheng. 2018. DeepTravel: a Neural Network Based Travel Time Estimation Model with Auxiliary Supervision. In *Proceedings of IJCAI*. 3655–3661.
[24] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *Proceedings of UAI*. 339–349.